# Distributed Computing for Autonomous On Board Planning and Sequence Validations

Dr. Pierre F. Maldague,
Dr. Leon Alkalai,
Dr. Savio Chau,
Dr. Kar-Ming Cheung,
Mr. David Tong,
and
Mr. Adans Y. Ko

Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
818-393-4813
adans.y.ko@jpl.nasa.gov

Abstract— We propose a new conceptual approach to system-level autonomy that exploits in a synergistic way recent breakthroughs in three specific areas:

1. Automatic generation of embeddable planning and validation software A traditional Activity Plan Generation Tool (APGEN) will be modified to a lightweight version of a Mission Planning and Validation Tool, that is suitable for insertion into the Command and Data Handler (C&DH) subsystem of an autonomous S/C. The embedded APGEN-lite will generate and validate science opportunities activities in real time onboard. In result, it will optimize the automatic delivery of science data to the gr

2. Integration of telecommunications forecaster and planning tools. To integrate a ground based telecom link analysis tool known as the telecommunications forecaster predictor (TFP) to S/C environment by taking advantage of APGEN-lite, that will make use of advanced telecom technologies such adaptive compression schemes.

3. Fault-tolerant assignment of computing tasks to multiple processors. Because it is responsible for its own planning and validation tasks, an autonomous S/C has much higher computing requirements than a conventional S/C. Therefore, a recently developed high-speed scalable fault tolerant distributed avionics architecture, which consists of two or more processors connected to multiple sensors, actuators, and science instruments by a high-speed, fault tolerant bus network.

## *1. INTRODUCTION*

1. Automatic generation of embeddable planning and validation software. Ground-based operations have traditionally relied on multi-mission planning and validation software , in particular the APGEN planner (Figure 1) used by JPL missions like Cassini, Odyssey, Deep Impact and MER, that can be easily adapted to different missions through the use of mission-specified text files. The breakthrough in this area is that APGEN can be modified with little effort so as to produce a lightweight version of itself. ("APGEN-lite") that is suitable for insertion into the C&DH subsystem of an autonomous S/C. As a result, we will validate a system that:

a) executes high-level requests from the ground
b) responds in real time to science opportunities
c) optimizes the automatic delivery of science data to the ground
d) dramatically reduces operations cost

2. Integration of telecommunications forecaster and planning tools. Traditionally telecom planning,

mission planning, and science planning are coordinated on the ground. The breakthrough is that we have integrated a ground-based link analysis tool known as the telecommunications forecaster predictor (TFP), currently used by missions like Deep Space 1, Cassini, Stardust. and Odyssey, with the APGEN planner to automate link prediction analysis. Our approach includes porting this integrated tool to the S/C environment by taking advantage of APGEN-lite and by developing an embedded version of the TFP tools that eliminates dependencies on the workstation environment such as graphics-oriented modeling tools in favor of self-contained support libraries suitable for on-board usage.

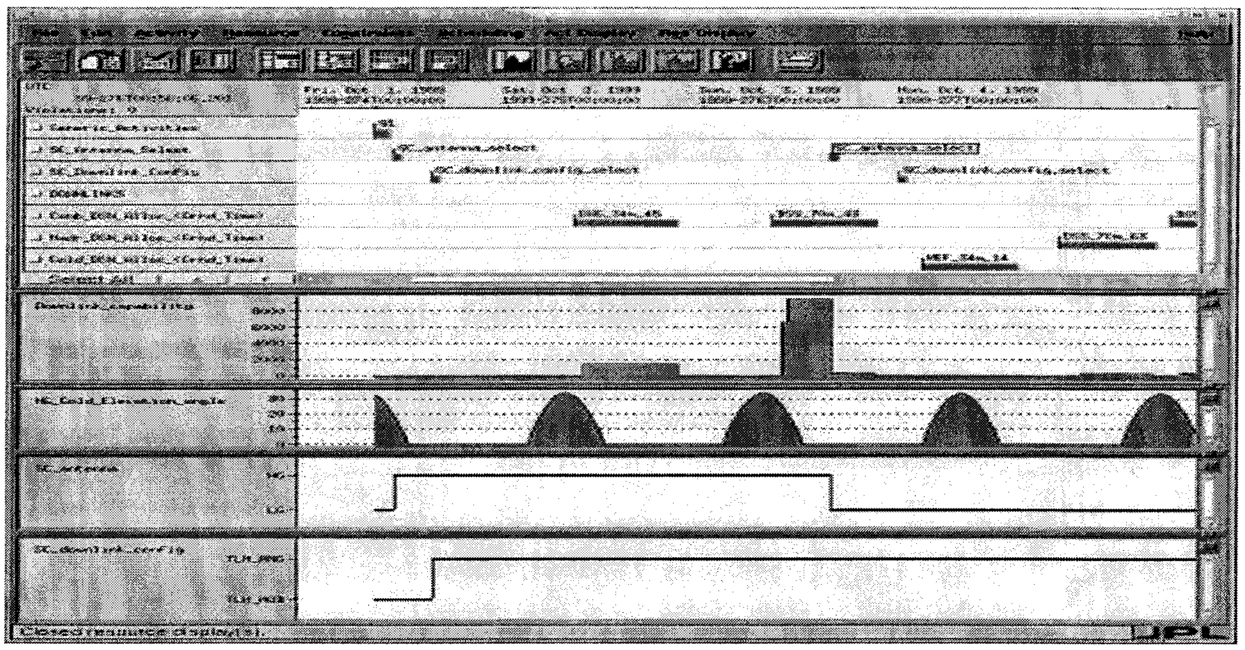


**Figure 1**: Apgen

3. Fault-tolerant assignment of computing tasks to multiple processors. Because it is responsible for its own planning and validation tasks, an autonomous S/C has much higher computing requirements than a conventional S/C. As a result, any hardware or software failures will have devastating effects on the mission. Therefore, the avionics architecture for autonomy has to be much more fault-tolerant than the traditional flight system design. The breakthrough that we exploit in our approach is a recently developed high-speed scalable fault tolerant distributed avionics architecture (Figure 2), which consists of two or more processors connected to multiple sensors, actuators, and science instruments by a high-speed, fault tolerant bus network. However, this architecture can be expanded to include up to 64 processors for missions that have more demanding computation needs Currently, in or real-time avionics testbed, we have a 5 node distributed system functionally executing VxWorks, and application software.

The key autonomy-enabling feature of our approach is the availability of each functional module and the protocol that orchestrates the interaction between the different modules, rather than the intrinsic "intelligence" of any individual module. The resulting benefit is that since our system architecture has already been validated, we have limited development risks to a single area, which is that of porting software to the S/C environment. Because this is a well-understood process, our approach minimizes software development risks.

The flight validation needed for this technology is how the system would behave upon real events in actual flight envirnoment. The system will record events and the response of the system. The correctness and response time will be examined to evaluate the effectiveness of the system. Although our approach is based on recent breakthroughs as outlined above, it is also conservative in the sense that it takes full advantage of multi-mission, reusable methodologies that are in use by a number of current space missions. The technology validation that is proposed here will therefore result in tools that are immediately usable by future missions thanks to the built-in adaptability of each technology module.

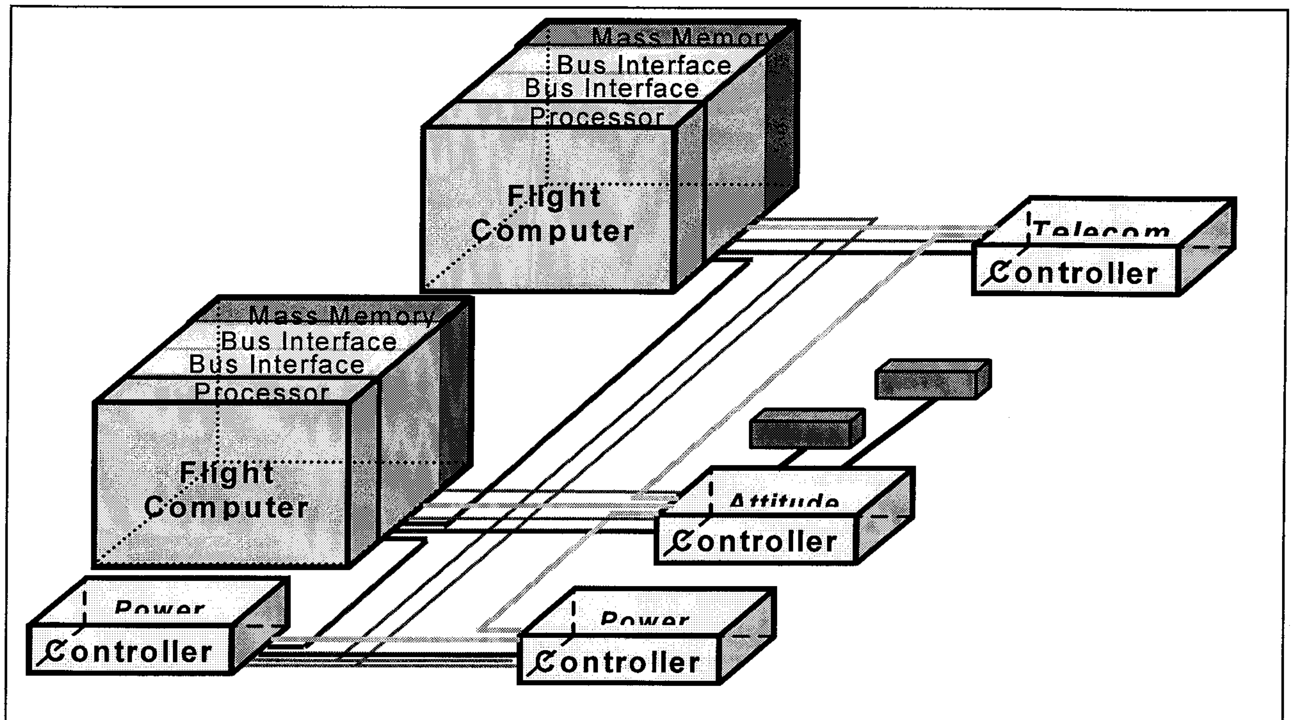


**Figure 2**: High-speed scalable fault tolerant distributed avionics architecture

## 2. *SYSTEM AUTONOMY*

We focus our approach on System Level Autonomy Software. The technology validation that is proposed here will result in tools that will immediately be usable by future missions thanks to the built-in adaptability of each technology module. Each module in our system has a well-defined responsibility:

- the scheduling module selects science activities from a list of things to do based on a well-defined priority scheme;
- the validation module verifies that the proposed schedule meets all flight and mission rules;
- the integrated telecom data rate prediction and planning module takes advantage of up-to-date ground and S/C information to maximize the rate of data return;
- the data compression module matches the level of compression to the available bandwidth;
- the re-planning module responds to faults in real time by selecting a recovery procedure that matches the signature of the fault.

### 2.1 Planning and Scheduling

One of the most challenging technical issues in the design of autonomous S/C is to design an on-board planning system that (a) accepts high-level goals, (b) reliably produces conflict-free plans, and (c) operates with predictable efficiency. There are elegant approaches to requirements (a) and (b), but the limited experience that has been obtained so far indicates that item (c), efficiency, is very difficult to achieve. We believe that progress will continue to be made in this area, and in fact two of us (Ko and Maldague) have been awarded a NRA 755 by NASA to explore the benefits of a mixed-initiative approach to planning that would combine the practical efficiency of APGEN with the elegant formalism of the Remote Agent planner used on-board DS1 as part of the RA Experiment.

However, the present approach does not require a complete solution to the planning problem just mentioned. Instead, we have taken a planning methodology that has been proven on the ground, and we have adapted it to the flight environment. In a nutshell, we assume that at every point in time the S/C has a valid sequence of commands in its sequence memory. As an opportunity arises to schedule a science observation or a downlink, an on-board scheduler performs the incremental planning work required to add the new activity to the sequence. The rationale for only scheduling one activity at a time is that the sequence that is held in memory at any one time is always a fixed-time sequence, in which the timing of each command is precisely known except possibly for the duration of the last command in the sequence (which may be uncertain up to a specified timeout.) Such a sequence can be modeled and validated with existing tools. If we tried to schedule more than one event-driven activity at a time, the set of all possible timing combinations resulting from the expansion of these activities into low-level commands would very quickly grow to unmanageable proportions; to our knowledge there is no methodology that currently could be used for validating such sequences.

Let us now explain in more detail how the approach works. Figure 3 below displays the essential components of the proposed system; in the next few paragraphs we will briefly describe the function of each component.

First we discuss the data flow through the system; then we will discuss the events that take place from the perspective of mission operations. High-level activities are generated on the ground and uplinked to the S/C via the gateway (steps 1 and 2 in Fig. 3). High-level activities generally include science observation candidates and downlink activities; let us concentrate on the former as an example. Observation candidates differ from traditional sequences in that they do not contain a specific start time; they only indicate a window of opportunity during which they should be scheduled. The candidates are therefore transferred to the Science subsystem (step 3 in the Fig. 3). As soon as the candidate's window overlaps the current planning period, the parameters of the observation are transferred to the planning subsystem (steps 4 and 5 in the Fig. 3) where whey will be tentatively scheduled by the "pretty good planner" (PGP), subject to approval by the rule checker (step 6 in the Fig. 3). If the observation passes the validity test, it is transferred to the Command and Data Handling subsystem (steps 7 and 8 in the Fig. 3), otherwise it is sent back to the Science subsystem (steps 7 and 3 in the Fig. 3) for possible scheduling later on.

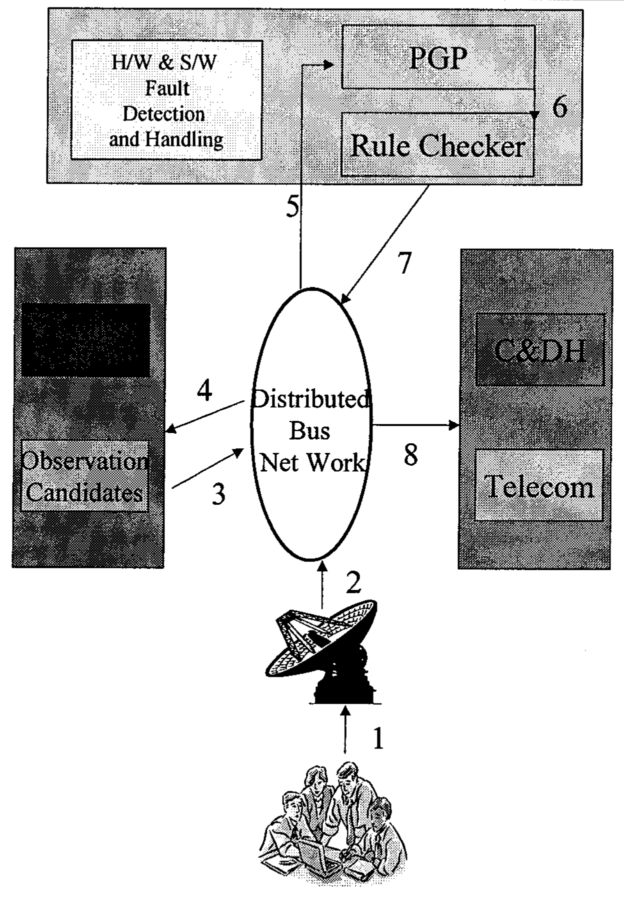


Fig. 3 High Level Commanding Data Flow

From an operations planning perspective, the first step is to establish a baseline plan that is essentially empty. It contains either a few or no science activities; its only purpose is to provide a fallback position that is known to be "safe" in case the fancy scheduling system described below fails to deliver a valid sequence. The baseline plan should contain a minimum of telecom and maintenance activity to guarantee that the ground does not lose contact with the S/C. The idea is to provide the mission with an opportunity to recover from whatever anomaly made it necessary to revert to the baseline plan.

The second step is to provide the S/C with a "list of things to do" from which the on-board planning system can extract its short-term plan. We will refer to this list as the List of Activity Candidates (LAC) in the following. The LAC contains two basic types of activities: science observations, which have been predefined by the science team, and downlink activities, which reflect the current DSN allocation plan. Unlike traditional sequences, both types of activities can have a non-trivial event-driven content. Science activities can be made conditional upon the result of one or more previous observations, such as the discovery of an unknown feature in a remote sensing observation. Downlink activities will be adjusted to reflect the amount of data currently held in on-board memory, the priorities of the various data sets to be downlinked, and the availability of compression algorithms to reduce the amount of data to be downlinked for each observation.

At this point, we need to explain the concept of an "incremental" sequence. At any time T, the S/C has in sequence memory a sequence of fixed-time instructions that represent the "currently committed sequence" or CCS. Initially, the CCS consists of the baseline sequence only. As time goes on, the CCS is built incrementally by adding activities from the LAC one by one. The task of selecting activities is assigned to the on-board scheduling engine, which we describe next.

The on-board scheduling engine is responsible for extracting potential activities from the LAC and scheduling them one-by-one based on predefined criteria similar to those currently used in ground-based planning. Typically, these criteria tell the scheduler (1) whether an activity from the LAC can be scheduled given what is known about the current state of S/C resources, and (2) at what time T' (> T) the activity should start. If an activity does not satisfy the criteria, the scheduler rejects it and extracts the next candidate from the LAC. The scheduler eventually finds an activity that passes the test. It then expands the activity in terms of well defined "expansion rules" that may take into account the current state of the S/C resources, and adds the resulting commands to the CCS, resulting in a new, proposed sequence (PS) (note that in order to convert the CCS into the PS, it may be necessary to remove or postpone as well as add commands; this is not a trivial issue but for simplicity we will ignore this refinement since it does not fundamentally alter the analysis.)

The planning engine that we actually propose to use is a lightweight version of the APGEN planner currently used in ground operations (see ref. [6]). We are not proposing to port APGEN to the flight environment. Instead, we want to use a technique that was developed recently to address performance problems that APGEN runs into when dealing with extremely detailed resource models, which require evaluating the state of the model a million times or more. The technique we came up with consists in using APGEN to convert its internal model (which was "compiled" from an external, mission-specific adaptation file) into executable code (currently C) that can be compiled and linked to provide a mission-specific, low-overhead version of APGEN itself which we call "APGEN lite". In recent tests, the lightweight version of APGEN runs between 10 and 100 times faster than the APGEN executable. Its memory footprint is about 20 times smaller than APGEN's, making it a suitable candidate for an on-board embedded planner.

Note that from the perspective of planning theory, APGEN as we propose to use it suffers from a basic flaw: the plans that it produces are not guaranteed to produce conflict-free sequences. The reason for this is that the basic scheduling algorithm bases its decision on a priori criteria, and does not do back-tracking in case the criteria fail to capture all the applicable flight rules and constraints. On the ground, this is not a serious problem for two reasons:

- the plans scheduled by APGEN are always double-checked by SEQGEN later on and any conflicts can be remedied manually
- as a *practical* matter, it is generally possible to come up with scheduling criteria that result in conflict-free plans 99% of the time. In the following, we will refer to a scheduler that achieves this level of performance as a "pretty good planner" (PGP)

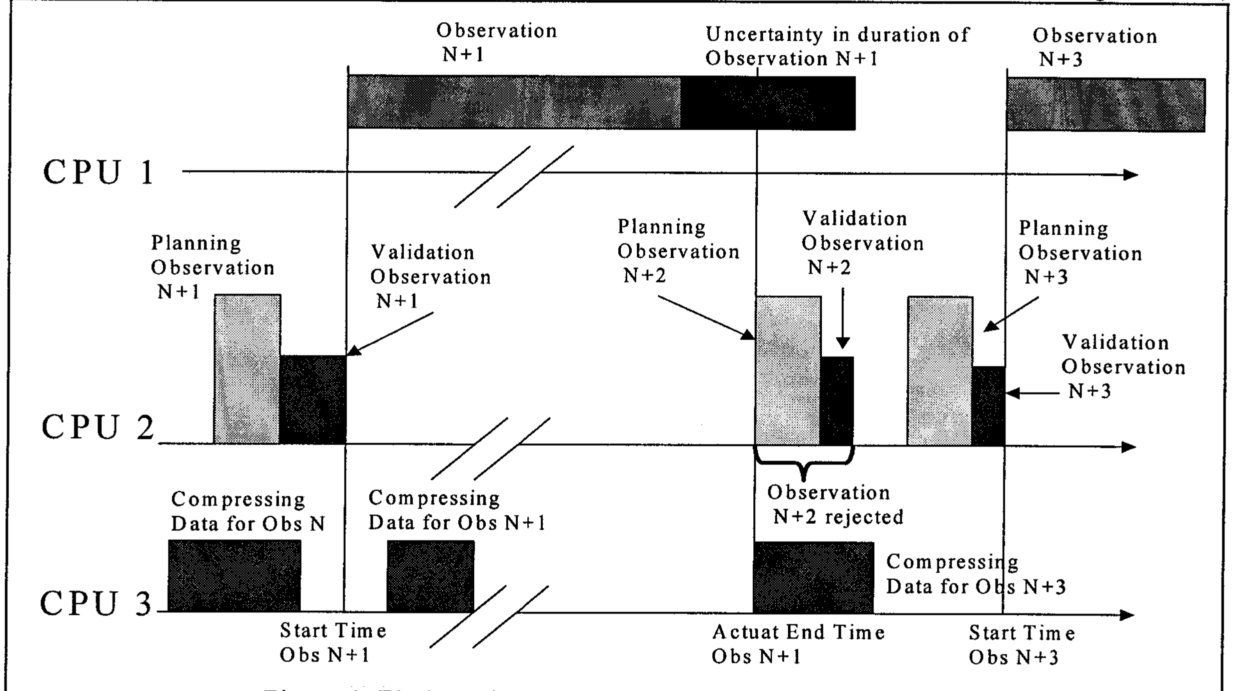
As far as planning technology is concerned, our approach is unabashedly empirical. If a fast, robust, provably correct algorithm was available for performing on-board planning, we would use it. In its absence, we will use a PGP to do the job. We state that a PGP can achieve a success rate of 99% for the following reasons. In recent, actual mission experience, the success rate of the scheduling algorithm is a direct consequence of the conceptual soundness of the APGEN adaptation that implements it. The first instance of APGEN scheduling in mission operations was going to be MPL ground operations before the mission was interrupted. A more recent example of such use of the scheduling algorithm is the automatic generation of aerobreaking sequences for Odyssey. In both cases, early implementations of the algorithm may have generated occasional conflicts, but after a few iterations it has always proved possible to come up with a reliable algorithm for conflict-free plan generation.

Just in case we happen to stumble into the 1% probability that the PGP will produce a plan with conflicts, we will adjoin to the PGP a validation step that provides the same functional level of constraint checking as SEQGEN provides when run as part of ground-based mission operations. Finally, to make sure that the probability of finding a conflict in the plan proposed by the PGP stays near 1%, we will exercise a ground-based scenario generator (GSG) to establish in a statistical sense that the PGP produces enough good plans for a given list of things to do. This will provide reassurance that most of the time the PGP will be able to come up with a valid sequence and the S/C won't be idle.

The next step consists in passing the PS from the sequencer to the validation engine, which performs a constraint check similar to that provided by SEQGEN in conventional flight missions. The engine that we actually propose to use is not SEQGEN (which was developed for a workstation environment and is not easily embeddable into S/C hardware) but a "lightweight" version of it similar to the on-board scheduler (the validation engine may end up being another incarnation of APGEN lite, but that is not important in the present discussion.) If the proposed sequence passes the validation test, the PS is turned into the new CCS. If it does not pass the test, the scheduler is so informed, and the scheduler is told to look for an alternative plan.
The above description concludes our brief description of one complete iteration of the process that we call "incremental planning". If we now step back to examine our approach from a higher-level perspective, we end up with something like figure 4 below.

The figure suggests that computational tasks can be distributed among several CPUs so as to provide the performance required to plan activities "just in time". Note that planning for Observation (N+2) cannot start until Observation (N+1) has completed, since the rules that need to be checked for Observation (N+2) may depend on time-dependent variables such as the position and velocity of the S/C relative to Earth, the amount of time left before the next downlink, or the precise amount of data available in the solid-state recorder. It is therefore important to provide as much computing power as possible to the scheduling and validation engines, since from a science point of view the S/C is essentially idle until they complete their work. The synergy between our approach to incremental planning and distributed computing is therefore very real.

Of course, the computational performance issue would be less important if there was no flexibility in the duration of observations, for then planning could proceed simultaneously with data taking. However, many observations would benefit from the timing flexibility that our approach provides. In addition, telecommunication issues such as the possibility of using Ka-band transmission depend in an essential way on physical phenomena such as Earth weather, which cannot be predicted far ahead of time. This is the second synergistic aspect of our proposal.


Observation N+1
Uncertainty in duration of Observation N+1
Observation N+3
CPU 1
Planning Observation N+1
Validation Observation N+1
Planning Observation N+2
Validation Observation N+2
Planning Observation N+3
Validation Observation N+3
CPU 2
Observation N+2 rejected
Compressing Data for Obs N
Compressing Data for Obs N+1
Compressing Data for Obs N+3
CPU 3
Start Time Obs N+1
Actuat End Time Obs N+1
Start Time Obs N+3
Figure 4: Timing Diagram for incremental Activity Planning and Execution

## 2.2 Onboard telecom automation scheme for intelligent communications

This scheme performs telecom prediction and sub-system level planning onboard the spacecraft. Telecom prediction involves estimating the communication bandwidth with a certain degree of confidence. Telecom planning interfaces with the system-level onboard mission planner and science planner schemes to maximize the overall science return of a mission, by:

1) providing link resource information to facilitate spacecraft activity planning;
2) setting up the link between the transmitting and the receiving ends;
3) receiving high-level science and communication goals and priority information to coordinate the onboard communications and signal processing.

Traditionally telecom planning, mission planning, and science planning are coordinated on the ground. The process is manually intensive and costly. The ongoing onboard automation thrust requires that these planning functions to be coordinated and executed on the spacecraft. Intelligent communications not only can maximize overall science return, it will reduce the spacecraft mass, power, recurring engineering costs and total life-cycle costs of the mission.

Our approach is to develop an intelligent telecom automation scheme that performs telecom prediction and subsystem-level planning onboard the spacecraft. The telecom predictor receives input from the system level autonomy software and other subsystem planners (navigation, attitude, power) to provide link resource information to facilitate spacecraft activity planning. The telecom planner sets up the link between the transmitting and the receiving ends. It also receives high-level science and communication goals and priority information to coordinate the onboard communications and signal processing schemes.

Link prediction estimates the bandwidth capability, usually in terms of supportable data rates, by taking into account the transmitter and receiver configurations, the spacecraft trajectory, onboard antenna pointing angles, and the communications geometry and media. Currently a ground-based link analysis tool known as the telecommunications forecaster predictor (TFP) exists. The TFP is a mature, proven, versatile link analysis tool based on MATLAB. The TFP tool is currently used by missions like Deep Space 1, Cassini, Mars Global Surveyor, Stardust, and Odyssey. To perform onboard link prediction, part of the TFP tool needs to migrate to the spacecraft. We propose to develop a lightweight version of TFP (TFP_Lite) in the C and VxWork environment that would fit well in the ST7 onboard architecture.

For onboard telecom planning, the future automated spacecraft and constellation of spacecrafts must be able to withstand a communications environment that is more dynamic and uncertain. The onboard telecom planner must be able to react to substantial bandwidth fluctuation in the face of dynamic resource utilization and rapid environmental changes, particularly pointing angles and available spacecraft power. Requests to update the predicted bandwidth can come at any time. The telecom planner has to be as optimal as possible, but must also be robust, anticipating run time changes.

One approach to effectively utilize a dynamic communication link is to execute multiple data rate changes within a pass. Changing data rate during a pass creates disruption to the physical channel, which in turn causes temporarily data dropout due to receiver out-of-lock. To maintain data continuity the spacecraft requires either transmitting zero-fill data or re-transmitting the data during the dropout period. The multiple data rate change strategy is currently being used by Cassini and DS1, and is executed with traditional ground developed sequence. Though manually intensive this operation mode increases data return by up to 70% in some passes. We propose to automate the multiple data rate change strategy onboard.

A system level approach is to develop an onboard automation scheme to coordinates several communications and signal processing technologies: link analysis, feature extraction, multi-resolution transform, progressive transmission, and data compression. This framework allows the use of onboard intelligence and automation to direct the signal processing techniques to maximize the overall science return of a single satellite as well as a constellation of satellites. We propose to use existing communications and signal processing components to the extent possible.

Traditional ground-based link prediction estimates the bandwidth capability, usually in terms of supportable data rates, by taking into account the transmitter and receiver configurations (telecom), the trajectories (navigation), the spacecraft antenna pointing (attitude), and the communication geometry and media (elevation angles and weather). In a ground-based scheme, many quantities are either assumed to be non-varying (for example, a specific type of ground antenna and a specific onboard configuration) or known for long periods in advance (for example, a "good enough" trajectory estimate that in turn determines station viewperiods and geometry). The difficulty arises in making estimates of items that are inter-dependent and often are determined in a ground-based sequence process, such as

1. spacecraft (and thus antenna) orientation as a function of activity or time in the mission;
2. available spacecraft power to telecom as a function of other subsystem and instrument activity;
3. competing uses of the available downlink power (such as telemetry and ranging or delta-DOR);
4. the ground stations allocated to the project as a function of time

Prediction for intelligent onboard telecom planning has to resolve the same dependencies. The onboard prediction tool would have the same kinds of quantities stored in its database, such as antenna gain vs. off point angle and the modulation index values for combinations of telemetry rates and ranging modulation. In addition, it has access to the equivalent of the trajectory to determine range and station elevation angles. Through interaction of the mission planner and science planner, telecom prediction would be provided with

1. candidate pointing strategies to translate into spacecraft antenna off point angles;
2. candidate spacecraft power profiles to translate into RF downlink power;
3. the amount of ranging or delta-DOR activity required for the mission phase

Development of the proposed telecom automation scheme includes figuring out the interactions (successive inputs and outputs) between it and the mission planner and science planner to handle the dependencies in an iterative fashion. Then the onboard system devises a suitable (though perhaps not optimum) telecom profile to go along with the attitude control profile, the power profile, and the instrument operation profile. The telecom predictor would provide a final output: the profile of station tracking time to accomplish the onboard determined telecom profile. An alternative approach might be that available station allocation profiles for the next planning cycle (developed in the existing or a new way on the ground) would be sent to the spacecraft. The onboard predictor would use these profiles, stored in a constraint file, in the same manner as it would for requirements on the amount of ranging data in the next planning cycle.

## 2.3 Fault-Tolerant Distributed Network for On-Board Autonomous Sequence Control

On-board autonomy poses new challenges to the avionics architecture. First, on-board autonomy has much higher computing requirements. A centralized flight computer system is not likely to be able to meet the demands. Second, since autonomy plays a more critical role in mission than the conventional flight software, any hardware or software failures that impact the autonomy will have devastating effects to the mission. Therefore, the avionics architecture for autonomy has to be much more fault-tolerant than the traditional flight system design.

In order to meet the demanding performance and fault tolerance requirements, a high-speed scalable distributed avionics architecture is proposed to support on-board autonomy. This avionics architecture consists of two processors connected to multiple sensors, actuators, and science instruments by a high-speed, fault tolerant bus network. However, this architecture can be expanded to include up to 64 processors for missions that have more demanding computation needs. The bus network consists of dual IEEE 1394 buses and two I2C buses. The IEEE 1394 bus is a multi-master bus that is capable to transfer data at 100, 200, or 400 Mbps. It can accommodate up to 64 nodes, each of which is either a computer or a microcontroller. The IEEE 1394 bus can guarantee on-time delivery of messages with the isochronous transaction and guarantees reliable delivery with the asynchronous transaction. The IEEE 1394 bus has adopted a tree topology and susceptible to branch node failure. Therefore, a redundant IEEE 1394 is used to backup the primary IEEE 1394 bus. The nodes of the two buses are connected in such a way that any node or link failure will not affect both buses. In addition, two I2C buses are used to facilitate the diagnostics and "repair" of the failures in the IEEE 1394 bus.

Every node has a primary role in the system related to the type of equipment that is controlled by the node. On the other, the processor or microcontroller in each node usually has more performance than required by its primary role. These performance margins can be used to share the workload of the primary flight computer, such as performing non-real time computations (e.g., sequence planner) for the autonomy. If it is necessary, more computers can be added to the bus network to further enhance the system performance. It is the plan of this proposal to investigate the technique to balance the workload dynamically, so that system resources can be used more efficiently.

Furthermore, this architecture has many fault tolerance provisions. These provisions will target at the general failure modes of the avionics system. In general, failure modes occurred in an avionics system can be categorized into three classes: crash, delay, and arbitrary failure. In the crash failure mode, a node will fail to respond to any input. The delay failure mode will respond to external input but the responses fail to meet the deadline. The arbitrary failure mode will also respond to external input but the response is erroneous. All these failure modes can be caused either by hardware, software, or the environment.

The crash failure mode can easily be detected by the total lack of response within a time limit. The strategy to recover from this kind of failure is to first reset the processor or microcontroller of the failed node. This would eliminate transient faults caused by software or environment (e.g., single event upset). If the reset fails, then the failure is most probably caused by hardware. The most effective remedy is to shut down the power of the failed node and redistribute its workload to other nodes. The delay failure mode can be detected by checking the delays of each response. If the delays would cause the execution of the sequence fail to meet deadline, then an alternative sequence will be invoked. If the alternative sequence also fails to meet the deadline, then the fault most likely has also occurred in hardware. Therefore, the workload of the failed node will be redistributed to other nodes. A similar recovery strategy can be used for the arbitrary mode, except that this failure mode has to be detected by on-board validation of the sequence.

One advantage of the IEEE 1394 bus is that the node failures in general will not cause a bus network failure since the physical layer of the network is kept alive by the power line in the cables. Therefore, the physical layer of the failed node can continue to pass on messages although its processor has failed. However, there are faults that would affect the physical layer in a node. When this type of faults happens, the tree topology of the IEEE 1394 bus will be destroyed. Therefore, the bus network needs a strategy to recover from network faults. This is explained in the following.

When a fault is detected in the primary IEEE 1394 bus, simple recovery procedures such as retry and bus reset will first be attempted. If the simple procedures cannot correct the problem, then all the workload will be transferred to the backup IEEE 1394 buses. Then, the system can have more time to diagnose the failed IEEE 1394 bus while the backup bus is performing the normal system operations. The diagnostic and "repair" of the failure are conducted with the assistance of the I2C bus. During the diagnostics, the root node of the IEEE 1394 bus will first send messages on the I2C bus to interrogate the health of all other nodes and the conditions of their connectivity. All nodes are supposed to respond to the root node's request via the I2C bus. If a node does not respond or if it indicates any physical connection failures, then the root node will send commands on the I2C bus to command the other nodes to reconfigure their links to bypass the failed node. After the failed node or connection is removed, the repaired IEEE 1394 bus will become the backup. Based on analysis, the bus network shown in Figure 1 can tolerate a minimum of three node or link failures.

## *3. CONCLSION*

Our approach will validate the following four areas:

1. Breadth and Generality of Autonomous High-Level Commanding. We will validate the on-board sequence planning and validation system by performing specific tasks and verifying that the performance targets have been met. The tasks chosen for this purpose will be:

**Long-term planning**: the on-board planning system will plan a one-month worth of prioritized science activities. Because our approach focuses on science observations that are event-driven, the result of this planning exercise will consists of a "plausible scenario" rather than an actual sequence to be executed on-board. The scenario will simulate the values of parameters that will only be known at execution time to come up with a complete sequence.

**Short-term planning**: the on-board system will perform incremental planning of science observations and telecom activities as described in the body of this proposal. The performance target for incremental planning will be consistent with the ST7 target of 5 seconds of planning time for science activities that do not affect critical times and 60 seconds of planning time for creating a plan in response to a new high-level request.

The on-board rule checker will be responsible for checking the flight rules, operational constraints, and resource contention in real time. The performance of the rule checker during normal operations will be monitored in terms of both its CPU time requirements and the accuracy of its conclusions.
2. Onboard Telecommunication Planning
We will conduct in flight validation of on-board downlink planning. The spacecraft will initiate communication activities to maximize the overall data return.

3. Fault tolerance and re-assignment of computing tasks to different CPUs. The ability of the fault detection and recovery subsystem to deal with hardware and software faults will be tested in a ground-based simulation environment (testbed) and verified in flight.

4. Adaptability to future missions. The methodology used in the course of implementing the present proposal will be documented following the same guidelines as those that apply to multi-mission ground data software. The documentation will include the actual adaptation files that were used to customize the planning, validation and telecom tools to the specific mission environment. It will also include the test design documents that will have been used to verify the performance of all subsystems. This process will pave the way for the re-use of our technology by the systems engineering teams of future missions.

## 4. *REFERENCES*

[1] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, "Design of a fault-tolerant COTS-based bus architecture," IEEE Trans. Reliability, vol. 48, pp. 351-359, Dec. 1999.

[2] S. N. Chau and M. Lang, "A Multi-Mission Testbed For Advanced Technologies," Innovative Approaches to Outer Planetary Exploration Workshop, 2001–2020, Lunar and Planetary Institute, Houston, Texas, February, 2001

[3] S. Chau et al, "The Implementation of a COTS Based Fault Tolerant Avionics Bus Architecture", in the Proceedings of the Aerospace 2000 Conference, Big Sky, Montana, Mar. 2000

[4] S. N. Chau, L. Alkalai, and A. T. Tai, "A multi-layer methodology for applying COTS in mission-critical systems," in Proceedings of the IEEE Workshop on Application-Specific Software Engineering and Technology (ASSET 2000), (Dallas, TX), pp. 70-76, Mar. 2000.

[5] T. Tai, S. N. Chau, and L. Alkalai, "COTS-based fault tolerance in deep space: Qualitative and quantitative analyses of a bus network architecture," in Proceedings of the 4th IEEE International Symposium on High Assurance Systems Engineering, (Washington, DC), pp. 97-104, Nov. 1999.

[6] Maldague, P., Ko, A.Y., Page, D.N. and Starbird, T.W., 1997, "APGEN: A Multi-Mission Semi-Automated Planning Tool", in First International Workshop on Planning and Scheduling for Space Exploration and Science.

[7] Maldague, P., Ko, A.Y., "JIT Planning: an Approach to Autonomous Scheduling for Space Missions", in March, 1999, IEEE Aerospace Conference, Aspen, Colorado

[8] K. Cheung, A. Makovsky, F. Taylor, "Telecommunications Analysis Service for DS1 Planning and Operations," submitted to IEEE Aerospace 2000 Conference, March 2000.

[9] Cheung, K.-M., "A Simple Algorithm for Automated High-Efficiency Tracking," TDA PR 42-130, April-June 1997, pp. 1-7, August 15, 1997.

[10] Cheung, K.-M., M. Belongie, and K. Tong, "End-to-End System Consideration of the Galileo Image Compression System," TDA PR 42-126, April-June 1996, pp. 1-11, August 15, 1996.

[11] Feria, Y., and K.-M. Cheung, "Seamless Data-Rate Change Using Punctured Convolutional Codes for Time-Varying Signal-to-Noise Ratios," TDA PR 42-120, October-December 1994, pp. 18-28, February 15, 1995.

[12] Statman, J. I., K.-M. Cheung, T. H. Chauvin, J. Rabkin, and M. L. Belongie, "Decoder Synchronization for Deep Space Missions," TDA PR 42-116, October-December 1993, pp. 121-127, February 15, 1994.

[13] K. Cheung and K. Tong, / Proposed Data Compression Schemes for the Galileo S-Band Contingency Mission," Proceedings of the 1993 Space and Earth Science Data Compression Workshop, Snowbird, Utah, pp. 99{110, April 2, 1993}.

[14] TFP: K. K. Tong and R. H. Tung, "A Multi-mission Deep Space Telecommunications Analysis Tool: Telecom Forecaster Predictor," IEEE Aerospace 2000, "Gateway to 21st Century Technology," Big Sky, Montana, March 18-25, 2000

## 5. *BIOGRAPHY*

***Dr. Savio N. Chau** received his Ph.D. in computer science from the University of California, Los Angeles in 1989. He is a senior system engineer and the task manager of the X2000 Future Deliveries Project at the Jet Propulsion Laboratory. He is currently developing scalable multi-mission avionics system architectures and investigating techniques to apply low-cost commercial bus standards in highly reliable long-life spacecraft. His research areas include scalable distributed system architecture, fault tolerance system design, architecture modeling, and rapid integration of intellectual properties on ASICs*

***Dr. Kar-Ming Cheung** is a Technical Group Supervisor in the Communications Systems Research Section (331) at JPL. His group provides telecom analysis support for JPL missions, and develops the operational telecom analysis and predict generation tools for current and future JPL missions and the DSN. He received NASA's Exceptional Service Medal for his work on Galileo's onboard image compression scheme. He was the Manager of the Network Signal Processing Work Area of NASA's Deep Space Network (DSN) Technology Program. He has authored or co-authored 6 journal papers and over 20 conference papers in the areas of error-correction coding, data compression, image processing, and telecom system operations. Since 1987 he has been with JPL where he is involved in research, development, production, operation, and management of advanced channel coding, source coding, synchronization, image restoration, and link analysis schemes. He got his B.S.E.E. degree from the University of Michigan, Ann Arbor in 1984, his M.S. degree and Ph.D. degree from California Institute of Technology in 1985 and 1987 respectively.*